The survival package

Terry Therneau

March 29, 2019

Chapter 1

Introduction

Work on the survival package began in 1985 in connection with the analysis of medical research data, without any realization at the time that the work would become a package. Eventually the software was placed on the Statlib repository hosted by Carnegie Mellon University. Multiple version were released in this fashion but I don't have a list of dates — version 2 was the first to make use of the `print` method that was introduced in 'New S' in 1988, which places that release somewhere in 1989. The library was eventually incorportated directly in S-Plus, and from there it became a standard part of R.

Looking back, I think that one of the primary reasons for the package's success is that all of the functions have been written to solve real analysis questions that arose from real data sets; theoretical issues were explored when necessary but they never played a leading role. As a statistician in a major medical center, the central focus of my department is to advance medicine; statistics is a tool to that end. This also highlights one of the deficiencies of the package: if a particular analysis question has not yet arisen in one of my studies then the survival package will also have nothing to say on the topic. Luckily there are many other R packages that build on or extend the survival package, and anyone working in the field, the author included, will use more packages than this one.

I certainly never foresaw that the library would become as popular as it has.

This vignette is an introduction to version 3.x of the survival package. I think of versions 1.x as the S-Plus era and 2.1 – 2.44 as maturation of the package in R. Version 3 had 4 major goals.

- Make mulit-state curves and models as easy to use as an ordinary Kaplan-Meier and Cox model.
- Deeper support for absolute risk models.
- More consitent support of robust variance estimates.
- Clean up various naming inconsistencies that have arisen over time.

With over 600 dependent packages in 2019, not counting Bioconductor, other guiding lights of the change are

- We can't do everything (so don't try).

- Allow other packages to build on this one. That means clear documentation of all of the objects that are produced, the use of simple objects (no S4 classes) that are easy to manipulate, and setting up many of the routines as a pair. For example `concordance` and `concordance.fit`; the former is the user front end and the latter does the actual work. Other package authors might want to access the lower level interface.

- Don't bugger it up!

This meant preserving the current argument names as much as possible. Appendix **??** summarizes changes that were made which are not backwards compatable.

The two other major changes are to collapse the many vignettes into this single large one, and the parallel creation of an actual book. This latter recognizes that the package needs more than a vignette. With the book's appearance this vignette can be more brief, essentially leaving out a lot of the theory.

Version 3 will not appear all at once, however; it will take some time to get all of the documentation sorted out in the way that we like.

1.1 Changes in version 3

Version 3.0 of the package was released in conjunction with a book. Writing the book, and in particular the examples, revealed some shortcomings in the design. In particular, there were some common concepts which had appeared piecemeal in more than one function, but not using the same keywords. Two particular areas are survival curves and multiple observations per subject.

Survival and cumulative hazard curves are genereated by the `survfit` function, either from raw data (survfit.formula), or a fitted Cox or parametric survival model (survfit.coxph, survfit.survreg). Two choices that appear are

1. If there are tied event times, to estimate the hazard using a straightforward increment of (number of events)/(number at risk), or make a correction for the ties. The simpler method is known variously as the Nelson, Aalen, Breslow, and Tsiatis estimate, along with hypenenated forms combining 2 or 3 of them. The same basic formula has been re-created in many contexts. One of the simpler corrections for ties is known as the Fleming-Harrington approximation when used with raw data, and the Efron when used in a Cox model.

2. The survival curve $S(t)$ can be estimated directly or as the exponential of the cumulative hazard estimate. The first of these is known as the Kaplan-Meier, cumulative incidence (CI), Aalen-Johansen, and Kalbfleisch-Prentice estimate, depending on context, the second as a Fleming-Harringtion, Breslow, or Efron estimate, again depending on context.

With respect to the two above, subtypes of the `survfit` routine have had either a `type` or `method` argument over the years which tried to capture both of these at the same time, and consequently have had a bewildering number of options, for example "fleming-harrington" in `survfit.formula` stood for the simple cumulative hazard estimate plus the exponential survival estimate, "fh2" specified the tie-corrected cumulative hazard plus exponential survival, while `survfit.coxph` used "breslow" and "efron" for the same two combinations. The updated routines

now have separate `stype` and `ctype` arguments. For the first 1= direct and 2=exponent of the cumulative hazard and for the second 1=simple and 2= corrected.

The Cox model is a special case in two ways: 1. the the way in which ties are treated in the likelihood should match the way they are treated in creating the hazard and 2. the direct estimate of survival can be very difficult to compute. The survival package's default is to use the `ctype` option which matches the ties option of the `coxph` call along with an exponential estimate of survival. This `ctype` choice preserves some useful properties of the martingale residuals.

A second issue is multiple observations per subject, and how those impact the computations. This leads to 3 common arguments of

- id: an identifier in each row of the data, which allows the routines to identify multiple rows for a subject
- cluster: identify correlated rows, which should be combined when creating the robust variance
- robust: TRUE or FALSE, to compute a robust variance.

These arguments have been inconsistent in the past, partly because of the sequential appearance of multiple use cases. The package started with only the simplest data form: one observation per subject, one endpoint. To this has been added

a. Multiple observations per subject
b. Multiple endpoints per subject
c. Multiple types of endpoints

Case (a) arises as a way to code time-dependent covariates, and in this case an `id` statement is not needed, and in fact you will get the same estimates and standard errors with or without it. (There will be a change in the counts of subjects who leave or enter an interval, since an observation pair (0, 10), (10, 20) for the same subject will not count as an exit (censor) at 10 plus another entry at 10.) If (b) is true then the robust variance is called for, one will want to have either a `cluster` argument or the `robust=TRUE` argument. In the coxph routine a `cluster(group)` term in the model statement can be used instead of the cluster argument, but this is no longer the preferred form. When (b) and (c) are true then the `id` statement is required in order to obtain a correct *estimate* of the result. This is also the case for (c) alone when subjects do not all start in the same state. For competing risks data — multiple endpoints, everyone starts in the same state, only one transition per subject — the `id` statement is not necessary nor (I think) is a robust variance.

When there is an `id` statement but no `cluster` or `robust` directive, then the programs will use (b) as a litmus test to decide between model based or robust variance, if possible. (There are edge cases where only one of the two has been implemented). If there is a `cluster` argument then `robust=TRUE` is assumed. If only a `robust=TRUE` argument is given then the code treats each line of data as independent.

Appendix A

Changes from version 2.44 to 3.0

Not all of these may be completed by 3.0, but this is the roadmap.

A.1 Survfit

The survfit object is changing. The primary change has to do with the starting time. When the package was first written in the late 1980s I made the decision to *not* include the initial point of the survival curve (time=0, S=1) as a part of the `time` and `survival` parts of the object; they could get tacked on by the plot function. With the addition of delayed start (`start.time` option) and then more importantly multi-state models, the starting point is not always a simple (0,1) pair, which led to the addition of new components to the objects to hold the additional information, and increasing if-then-else logic in the downstream routines that process the curves. In v3 the first point is now bundled in as part of the time, surv, pstate, std.err, and etc components.

Two functions survfit23 and survfit32 convert between the version 2 `survfit` object and new `survfit3` forms. This has allowed us make any changes incrementally.

Other changes are

- Common arguments of id, cluster, and influence
- The routines now produce both the estimated survival and the estimated cumulative hazard, along with their errors
- Some code paths produced std(S) and some std(log(S)), the object now contains a `log.se` flag telling which. (Before, downstream routines just "had to know").
- an explicit "v3" flag in the object

A.2 Coxph

The multi-state objects include a `states` vector, which is a simple list of the state names. The `cmap` component is an integer matrix with one row for each term in the model and one column for each transition. Each element indexes a position in the coefficient vector and variance matrix.

- Column labels are of the form 1:2, which denotes a transition from `state[1]` to `state[2]`.
- If a particluar term in the data, "age" say, was not part of the model for a particular transition then a 0 will appear in that position of `cmap`.
- If two transitions share a common coefficient, both those element of `cmap` will point to the same location.
- The first row of `cmap`, labeled "stratum", is numbered separately and partitions the transtions into disjoint strata.